Three-Way DEDICOM for Relational Learning

Maximilian Nickel
University of Munich
nickel@cip.ifi.lmu.de

Volker Tresp
Siemens AG
volker.tresp@siemens.com

1 Introduction

Relational learning is finding an increasing number of applications in such diverse areas as social network modeling, the semantic web, bioinformatics and artificial intelligence. At the same time, tensors and their decompositions are widely used in fields like psycho- or chemometrics and more recently have also been applied to data mining and machine learning problems, e.g. for modeling temporal effects in social networks such as the WWW. The goal of this paper is to explore some applications of tensor decompositions to relational learning. The main motivation is that domains with multiple relations of any order can be expressed straightforwardly as a higher-order tensor. Here we have a slightly different focus: we demonstrate the usefulness of a special kind of tensor decomposition, i.e. 3-way DEDICOM [5], for modeling domains with multiple binary relations in the tasks of instance matching and collective classification. We use the semantic web's RDF formalism where relations are modeled as RDF triples of the form `(subject, predicate, object)` and where a predicate either models the relationship between two entities or between an entity and an attribute value. Relational data are modelled as a 3-way tensor $\mathcal{X}$, where two modes are identically formed by the concatenated entities of the domain[1] and the multiple predicates define the third mode. A tensor entry $\mathcal{X}_{ikj} \neq 0$ denotes the fact that there exists a triple `(i-th entity, j-th predicate, k-th entity)`. An attribute value will be modeled differently using kernels as discussed in the following section.

2 Methods and theoretical aspects

Relational learning is concerned with domains where the entities are connected by multiple relations. In such a domain it can be of great benefit when the learning algorithm is able to propagate information across multiple relations. For instance, consider the task of predicting the party membership of a president resp. vice president of the United States of America. Without any additional information, this can be done quite accurately when the party of the president's vice president (or vice versa) is known, since both persons have mostly been members of the same party. Figure 1(a) shows a visualization of the set of (vice) presidents, their party-membership and their relations. To benefit from relational modelling, a learning algorithm should be able to propagate the party information across the (vice-)presidency relations when the party membership of a particular person in the population is unknown.

As described earlier, two modes consist of the domain's entities and the predicates form the third. In order for a tensor decomposition to automatically detect the discussed correlations across relations, it is important that *subjects* and *objects* of different relations are recognized to be identical when they refer to the same entity, as only then chains like `(partyX partyOf personB) (personB presidentOf personA)` can be detected. Furthermore, the relations are not necessarily symmetric and consequently the predicate-slices aren't symmetric either. Motivated by these considerations we chose the 3-way DEDICOM decomposition, as it has been developed for this situation and meets all requirements.

[1] Please note that we don't assume homogeneous domains, thus the entities of one mode can be instances of multiple classes like persons, items, places, etc.

Using DEDICOM, canonical relational learning tasks can be approached as following. Let $\mathcal{X}_k$ be the tensor slice that describes the k-th predicate and m be the number of predicates. The 3-way DEDICOM decomposition of $\mathcal{X}$ is given by $\mathcal{X}_k \approx AD_kRD_kA^T$, for $k = 1, \ldots, m$. The entity-concept matrix A contains the latent components, R models the interactions, and the diagonal D_k models the participation of latent components in the k-th slice. *Entity resolution* can be performed by clustering entities on the entity-concept matrix A, while *link prediction* can be done by evaluating the entries of the appropriate reconstructed predicate-slice $AD_kRD_kA^T$. *Collective Classification* can be regarded as a subtask of link prediction, as the class of an entity can by modelled by introducing a class relation. Thus it can also be solved by reconstructing the appropriate predicate-slice.

In the discussion so far we assumed that the objects of a predicate are entities. Now we consider the case that the object is an attribute value. Attributes of an entity are usually continuous (e.g. age) or discrete (e.g. textual description) variables. Conceptionally, one could use some form of data processing techniques like binning or n-grams and treat the modified attribute values in the same way as entities are treated in DEDICOM. However, this would result in a significantly increased dimensionality and sparsity of the tensor, since e.g. semantical considerations would require that different predicates use different bins and because attributes like proper names can have very unique values. Instead, we propose to use kernel matrices for attributes. The rationale in doing so is that for attributes one isn't interested in information propagation, but in computing the similarity of entities by the means of these attributes. Expressing this similarity is exactly the purpose of a kernel matrix. Since a kernel matrix is a $entities \times entities$ matrix it can also be used as a predicate-slice in the tensor. We propose to create an entity-attribute matrix E where the rows correspond to the entities that are present in the entity-modes and the columns correspond to one or more attributes of these entities. Then, the predicate slice K for these attributes can be computed by an appropriate kernel, in the simplest case by the linear kernel $K = E^TE$. To ensure that none of the attribute-kernels dominates the decomposition, it might be necessary to normalize or weight these kernel matrices appropriately.

3 Experiments

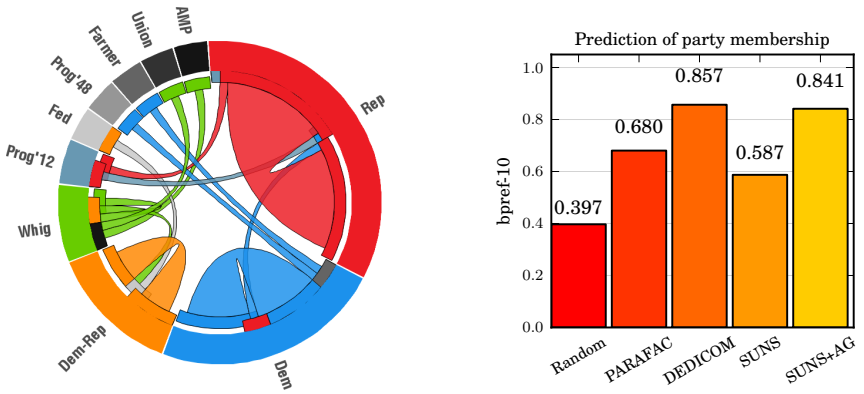
In this section we present two applications of DEDICOM on relational datasets. DEDICOM has been implemented as the ASALSAN [1] algorithm on top of the Tensor Toolbox [2].

3.1 Information Propagation

The first experiment demonstrates the benefits of information propagation as discussed in Section 2. We retrieved the names of all presidents and vice presidents of the United States from DBpedia, in combination with their party membership and the presidentOf/vicePresidentOf information. From this data we constructed a $93 \times 93 \times 3$ tensor, where the entity modes correspond to the joint set of persons and parties and the third mode to the three predicates. The goal of the experiment is to predict the party membership for each person and thus the problem can be regarded as a collective classification task. Please note that the dataset contains no further information that could help at predicting the correct party. Consequently, a machine learning algorithm should only be successful on this dataset when it is able to propagate information across the (vice-)president relation. To evaluate the algorithms on this data we've conducted leave-one-out cross validation by iterating over all persons and deleting the party-membership information of the person of interest. In the case of DEDICOM we performed a rank-2 decomposition, ranked all parties by their entries in the reconstructed party-membership-slice $AD_kRD_kA^T$ and recorded the bpref-10 score.

Figure 1(b) shows the results of DEDICOM compared to the PARAFAC decomposition on the same tensor. Moreover, we have included SUNS [6], a relational learning approach for large scale data. However, SUNS isn't able to propagate information across different relations. Aggregated SUNS (SUNS+AG), which mimics information propagation, improves the SUNS model significantly.[2] The results of DEDICOM outperform both PARAFAC and SUNS and thus show clearly the usefulness of our approach for domains where information propagation is required.

[2]In this context aggregation means that the party membership information of the related person has been added as a new attribute to each statistical unit



(a) (b)

Figure 1: Data visualization and experiments for the (vice-) presidents of the United States and their party associations. The size of the ring segments in Figure 1(a) indicate how many presidents and vice presidents originate from the particular party. An arc indicates a `presidentOf` relation and the size of an arc indicates how often this relation occurs between the connected segments.

3.2 Instance Matching

Instance matching is a term that is most frequently used in the semantic web community and refers to the task of identifying the entity descriptions (also called instances) from heterogeneous data sources that refer to identical real-world entities. Therefore, it can be regarded as an entity resolution problem. However, classical entity resolution is usually concerned with identifying entities that have different value representations, e.g. "M. Gandhi" or "Mahatma Gandhi", while it is assumed that the data adheres to one logical and structural representation, e.g. that the entities of type Person have an attribute birth place. This isn't the case for instance matching. Consider two data sources A and B. Source B might specify the birth place as a relation instead of an attribute or source A might link brothers and sisters by a siblings relation while source B specifies only a relation to the parents for these entities. Therefore, instance matching is an interesting problem for relational learning and is at the same time an important practical problem, for example at linking data in the semantic web.

To evaluate the applicability of DEDICOM to instance matching we used the IIMB 2009 benchmark suite.[3] The benchmark has been created by collecting data like movies and actors from the Internet Movie Database (IMDB). In particular, the data consists of 5 entity classes (Movie, Actor, PlaceOfBirth, Cast, AKA), 4 relations (BornIn, InterpretedBy, HasCast, AlsoKnownAs), 13 attributes (e.g. Name, Genre) and 302 entities. This reference data has been subjected to various transformations like introducing typographical errors or changing the structure of the model, what ultimately lead to 70 different data modifications in total. In the following we will refer to these modifications as test cases. The objective of the benchmark is to match the entities of each test case to the entities of the reference data. For a thorough discussion of the benchmark please refer to [4]. We expect that DEDICOM can produce good results in this problem domain, as the entity-concept matrix A is computed by minimizing the global reconstruction error across all modes and thus should also detect correlations across multiple predicates. Moreover, as the decomposition aggregates the predicates simultaneously to the entity-concept mapping, we anticipate that the entity-concept space is a meaningful representation for entity resolution even when the structural or logical representations of similar predicates are diverse.

In particular, we've applied the following procedure for each test case: First we created *one joint* tensor $\mathcal{X}$ for the test case and the reference data, such that the entities and predicates of both datasets form the entity modes or predicate slices, and set $\mathcal{X}_{ikj} = 1$ for each existing triple `(i-th entity, j-th relation, k-th entity)`. Furthermore, for selected predicates like `rdfs:label` or `imdb:name` we created kernel matrices as predicate-slices.[4] Therefore, we first built a TF-IDF

[3] available at http://www.instancematching.org/iimb2009

[4] In the present case, we've manually specified which predicate should be treated by string kernels. However, this could easily by automated by looking at the datatype of the object value

		Precision	Recall	F-Measure
Value Transformations (01-18)	DEDICOM	**0.968**	0.888	**0.926**
Typographical errors, Standard modifications	H-Match	0.928	**0.89**	0.908
Structural Transformations (19-37)	DEDICOM	**0.987**	**0.878**	**0.929**
Values deletion & separation, Depth modification	H-Match	0.885	0.862	0.872
Mixed Transformations (38-53)	DEDICOM	**0.982**	0.887	**0.933**
Value and structural transformations	H-Match	0.922	**0.915**	0.918
Logical Transformations (54-70)	DEDICOM	0.977	0.777	0.866
Different (sub)classes & instantiations, Implicit val.	H-Match	-	-	-

Table 1: Instance matching benchmark results of DEDICOM and HMatch. The best results for each measure are printed bold. The results of HMatch for the test cases 54-70 are missing, as the reference implementation wasn't able to parse the test files.

matrix T_i from the bigrams of the object values of the selected predicate. Afterwards we computed the predicate-slice for this predicate by the linear kernel $K_i = T_i^T T_i$. Varying with the particular test-case, this resulted in a tensor-size of approximately $740 \times 740 \times 20$. Following the construction of $\mathcal{X}$, we've applied DEDICOM such that $\mathcal{X}_k \approx AD_kRD_kA^T$. In order to match the entities of both data representations, we computed the nearest neighbour r_i of the reference data for each entity t_j of the test case in the concept-space A. Since equality is a symmetric relation we required that t_j is also the nearest neighbour of r_i. If this should be the case, we conclude that r_i and t_j refer to the same entity.

As a reference algorithm we used H-Match [3], a specialized instance and ontology matching algorithm that is able to use linguistic, contextual and structural features as well as reasoning to match instances and also gave good results at the 2009 OAEI Instance Matching Track. Table 1 shows that the DEDICOM approach for relational instance matching is very competitive. This is especially notable as we didn't incorporate any problem specific engineering, like WordNet similarity kernels.

4 Conclusion

In this short overview we have shown how the 3-way DEDICOM tensor decomposition can be applied to relational learning problems and how kernels can be utilized to handle attributes. In particular we have shown the benefits of our approach for information propagation and instance matching in multi-relational domains and achieved very promising predictive results using a custom dataset for collective classification and a benchmark suite for instance matching. To obtain highly scalable solutions we are currently investigating distributed versions of DEDICOM as well as exploiting constraints like typed relations.

References

[1] Brett W. Bader, Richard A. Harshman, and Tamara G. Kolda. Temporal analysis of semantic graphs using ASALSAN. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 33–42, Omaha, NE, USA, 2007.

[2] Brett W. Bader and Tamara G. Kolda. MATLAB tensor toolbox version 2.4, March 2010.

[3] S. Castano, A. Ferrara, and S. Montanelli. H-MATCH: an algorithm for dynamically matching ontologies in peer-based systems. In *Proc. of the 1st VLDB Int. Workshop on Semantic Web and Databases (SWDB 2003)*, Berlin, Germany, September 2003.

[4] Alfio Ferrara, Davide Lorusso, Stefano Montanelli, and Gaia Varese. Towards a benchmark for instance matching. In Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, and Heiner Stuckenschmidt, editors, *Ontology Matching (OM 2008)*, volume 431 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[5] R. A Harshman. Models for analysis of asymmetrical relationships among n objects or stimuli. In *First Joint Meeting of the Psychometric Society and the Society for Mathematical Psychology, McMaster University, Hamilton, Ontario, August*, 1978.

[6] Yi Huang, Volker Tresp, Markus Bundschus, and Achim Rettinger. Multivariate structured prediction for learning on semantic web. 2010.